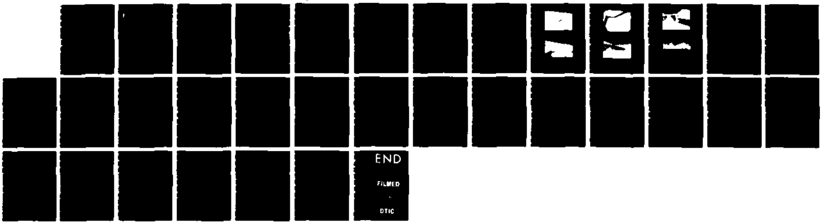
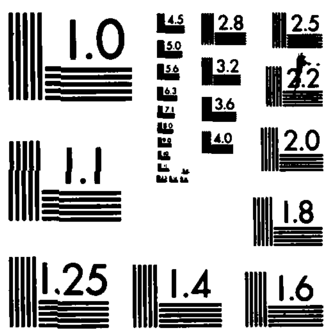
AD-A159 316 REAL-TIME DIGITAL TERRAIN IMAGE GENERATOR(U) TECHNOLOGY SERVICE CORP SANTA MONICA CA R CLARK ET AL. JUL 85 HEL-TM-10-85 DAAK11-83-C-0009 1/1

UNCLASSIFIED F/G 5/5 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A159 316

AD

Technical Memorandum 10-85

REAL-TIME DIGITAL TERRAIN IMAGE GENERATOR

Richard Clark
Richard Pferdner

July 1985
AMCMS Code 612716.H700011

DTIC
ELECTE
SEP 12 1985
S D
A

Approved for public release;
distribution unlimited.

U. S. ARMY HUMAN ENGINEERING LABORATORY
Aberdeen Proving Ground, Maryland

DTIC FILE COPY

85 9 12 002

Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Use of trade names in this report does not constitute an official endorsement or approval of the use of such commercial products.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-A159 316

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Memorandum 10-85	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) REAL-TIME DIGITAL TERRAIN IMAGE GENERATOR		5. TYPE OF REPORT & PERIOD COVERED Final 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard Clark Richard Pferdner		8. CONTRACT OR GRANT NUMBER(s) DAAK11-83-C-0009
9. PERFORMING ORGANIZATION NAME AND ADDRESS Technology Service Corporation 2950 31st Street Santa Monica, CA 90405		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS AMCMS Code 612716.H700011
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Human Engineering Laboratory Aberdeen Proving Ground, MD 21005-5001		12. REPORT DATE July 1985 13. NUMBER OF PAGES 30
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image Generation	Real-Time Simulation
Flight Simulator	Visual Simulation
HELICON	Helicopter Visual Flight Simulation
Computer Image Generation	Out-The-Window Simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Technology Service Corporation has created a real-time digital terrain image generator. The system, HELICON, has been installed and integrated into the Human Engineering Laboratory's (HEL) Human Factors Research Simulator (HFRS) system. HELICON receives location and attitude data in real time from one of the crew station components of the HFRS and provides stylized representations of terrain as would be seen from the crew station. The HELICON system exceeds most of the contract's performance

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

requirements, like gaming area size, flight capability, maximum viewing distance, and number of moving objects. It creates computer-generated imagery in full color at the rate of 15 frames per second (fps), with a density of 50 objects per square mile (mi^2) and an image resolution of 512 x 512.

AMCMS Code 612716.H700011 Technical Memorandum 10-85

REAL-TIME DIGITAL TERRAIN IMAGE GENERATOR

Richard Clark
Richard Pferdner

Point of Contact:

Richard S. Camden
Gordon L. Herald

July 1985

Technology Service Corporation
2950 31st Street
Santa Monica, CA 90405

APPROVED: John D. Weisz
JOHN D. WEISZ
Director
U.S. Army Human Engineering Laboratory

Approved for public release;
distribution unlimited.

U.S. ARMY HUMAN ENGINEERING LABORATORY
Aberdeen Proving Ground, Maryland 21005-5001

CONTENTS

DTIC
COPY
INSPECTED

Distribution/
Availability Codes
Dist
A-1

REAL-TIME DIGITAL TERRAIN IMAGE GENERATOR

INTRODUCTION

This is Technology Service Corporation's (TSC) final report on contract DAAK11-83-C-0009. The requirements of this project were to design, develop, and integrate a real-time digital terrain image generator. The system created by TSC under this contract, called HELICON, has now been installed at the U.S. Army Human Engineering Laboratory (USAHEL). The HELICON system has been installed and integrated into the USAHEL's Human Factors Research Simulator (HFRS) system which includes a LINK/GAT motion-based helicopter simulator, an advanced helicopter crew station research simulator, and an air defense crew station mock-up. HELICON receives location and attitude data in real time from one of the crew station components of the HFRS and provides stylized representations of terrain as would be seen from the crew station. Figures 1-3 exemplify the simulator's visual imagery for several different flight conditions.

Background

The USAHEL is the Army Materiel Command's lead laboratory for human factors engineering. Increasingly sophisticated equipment and soldier interfaces require the laboratory to perform effective investigations into complex soldier-machine interfaces. To enable the laboratory to perform effective research, the in-house interactive research simulation facility is being upgraded and enhanced. In particular, advances in display technology and the proposed application of new technologies to military equipment pose new human factors engineering challenges.

USAHEL plans future research investigations of human factors aspects of helicopter and vehicle crew performance using integrated displays of out-the-window views overlaid with computer weapon system and vehicle performance displays. Crew interaction with computer-produced terrain images will function as the workload driver imposing a maneuver task upon the pilot or vehicle driver. This level of terrain image generating capability is currently available only through very expensive visual systems built primarily for flight training.

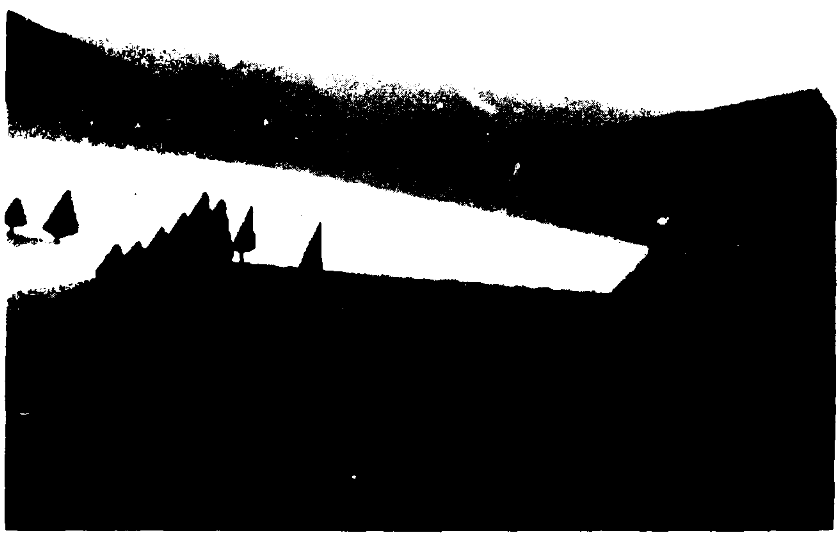
To provide this capability for use within the laboratory, USAHEL investigated state-of-the-art display processing systems and projected that commercially available systems probably could be adapted to produce a real-time interactive, scene-generating capability. Absolute fidelity and accuracy to the real world were not required because of the intended use of the system. Rather, the objective was to obtain a relatively inexpensive image-generating system that will run in real time, and produce a visual terrain image that is a tradeoff among scene realism, density, and level of detail. An innovative approach was required which used proven, commercially available display equipment interfaced with the existing USAHEL simulation facility.

a. Helicopter approaching hills and river scene, about 100-ft altitude.

b. Helicopter executing left turn, about 100-ft altitude.

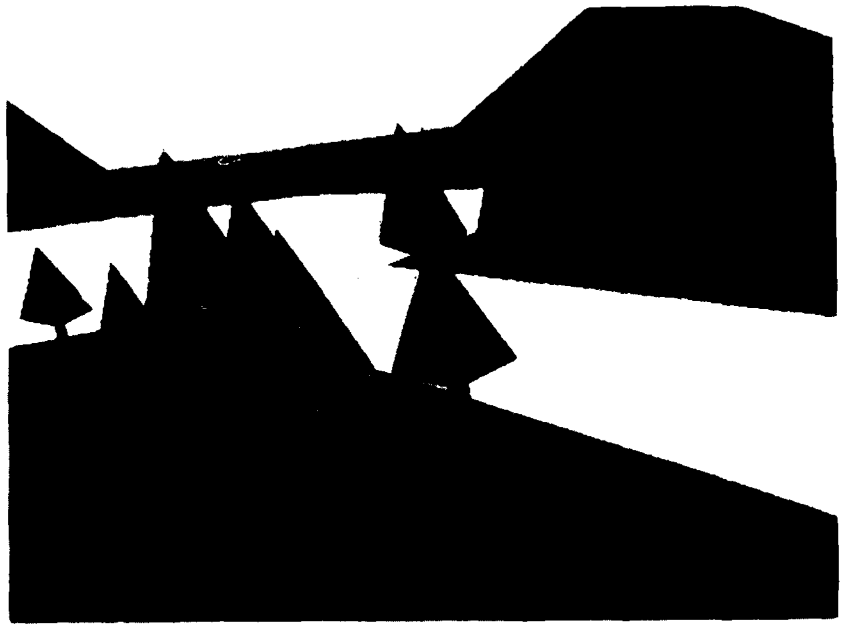
Figure 1. Examples of HELICON's visual output.

a. Helicopter in steep downward pitched flight.



b. Helicopter in nap-of-the-earth flight and approaching lake.

Figure 2. Examples of HELICON's visual output.



a. Helicopter approaching tree grove.

b. Helicopter hovering over lake.

Figure 3. Examples of HELICON's visual output.

The overall approach to creating the system was to develop a concept design, determine the appropriate hardware for the project, arrive at detailed designs for the system and algorithms, and develop the software.

SYSTEM SPECIFICATIONS

In this section, we have compared the system requirements called for in the contract to the resulting capabilities of the installed system. The items are grouped into five categories: data base, flight capability, system performance, special features, and software quality.

Data Base

The data base contains terrain features, or objects, located within a square gaming area.

Terrain Features

A terrain feature is a representation of a natural or man-made object. The system was required to have two-dimensional (2D) terrain features of roads and rivers, and 3D terrain features of trees, buildings, and hills. The system delivered provides all the required features, as well as lakes and fields.

Gaming Area

The system was required to contain a model of a gaming area measuring 5 x 5 mi, and the gaming area data base was required to be modifiable. The gaming area delivered is 5 x 5 mi, capable of expansion to 25 x 25 mi. A set of software tools was developed for modifying the existing gaming area data base and creating new data bases.

Maximum Feature Height

The system was required to have a maximum feature height of at least 200 ft. The system delivered has a maximum feature height of 500 ft.

Flight Capability

The flight capability is defined by the maximum and minimum altitude, roll, pitch, yaw, and speed.

Altitude

The altitude range required for the system was 0 to 500 ft, with an accuracy of 2 ft. The altitude range delivered is 0 to 1000 ft, with an accuracy of 2 in.

Roll, Pitch, and Yaw

The roll required was ±30°; the roll delivered is ±180°. The pitch required was ±6°; the pitch delivered is ±45°. The yaw required was ±10°; the yaw delivered is ±180°. The accuracy requirement for roll, pitch, and yaw was 1°; the accuracy delivered is better than 0.001°.

Speed

The system requirement for speed was 0 to 100 knots, with an accuracy of 1 knot. The speed delivered has no upper limit, and the accuracy is better than 0.001 knots.

System Performance

System performance is defined by object density, field of view (FOV), maximum viewing distance, update rate, image resolution, occultation, hidden surface removal, and number of moving objects.

Object Density

Object density is the number of terrain features per square mile (mi^2). The system was required to have 25 features/mi^2, uniformly distributed, in all but one 1 mi^2 area, which was to have a higher density of 50 features. The delivered system exceeds this requirement, offering 50 objects/mi^2 throughout the data base and not being limited to uniform distribution.

Field of View

The system was required to have an FOV of 40° horizontal and 30° vertical. The vertical angle requirement was later changed to 40° to simplify the clipping of polygons to the display window in the graphics hardware. The system delivered has an FOV of 40° horizontal and vertical.

Maximum Viewing Distance

The maximum viewing distance required of the system was 3000 ft along the centerline of the FOV from the viewing location. The maximum viewing distance delivered is 5000 ft. The distance at which an object comes into view is set for each object such that smaller objects, like trees, come into view at a shorter distance than larger objects, like hills. This setup reduces processing load and increases system performance.

Antialiasing: This capability reduces or eliminates aliasing artifacts such as stairstepping and crawling edges. Although we have reduced the severity of aliasing in the HELICON system by carefully selecting the color palette, we can further reduce these artifacts significantly by using an algorithm to blend adjacent colors at selected polygon edges and at the horizon line. We have identified the Gupta-Sproull algorithm (Gupta & Sproull, 1981) as being excellent for this purpose.

Level of detail: This capability would require storing object representations at several levels of detail and selecting the level for display based on the distance from viewer to object. This enhancement would reduce the average number of displayed polygons per object, since the most complex object representation would be displayed only when the object is close to the viewer. The result would be an increase in the polygon capacity of the system.

Flying objects: This enhancement would involve adding an altitude component to the moving object paths. To ensure proper hidden-surface removal, the flight paths might need to be restricted as to how close they could get to stationary objects.

More colors: The number of usable colors in the palette can be expanded from 256 to 1024 by changing the mapping of two output image memory bits. This enhancement would be necessary to add a sun shading capability.

Sun shading: To produce sun shading, the relative brightness of each polygonal surface in the data base is computed using the surface's angle relative to the sun plus an ambient light factor. The brightness computation would assume diffuse reflection (i.e., Lambert's cosine law). Sun shading processing would be done off line to avoid slowing the real-time software, which means sun position would be fixed during a simulated flight.

Shadows: Software could be written to compute for each object a shadow polygon on the ground plane. We estimate that the addition of shadows to the data base would require reducing the object density by 10 percent to 30 percent to maintain an update rate of 15 fps. Therefore, we foresee shadows being used selectively to add realism in certain scenarios only.

Light points: We have developed an approach to displaying antialiased light points that avoids having them appear "pixelated." This capability could be added.

Graphical data base editor: A graphical editor would provide the ability to display on a color monitor all or part of a data base and to add to or alter it using an interactive device such as a joystick or mouse. We estimate that use of such an editor would speed the creation of data bases for the HELICON simulator by as much as a factor of 10. (This addition would be the most involved of the proposed short-term enhancements.)

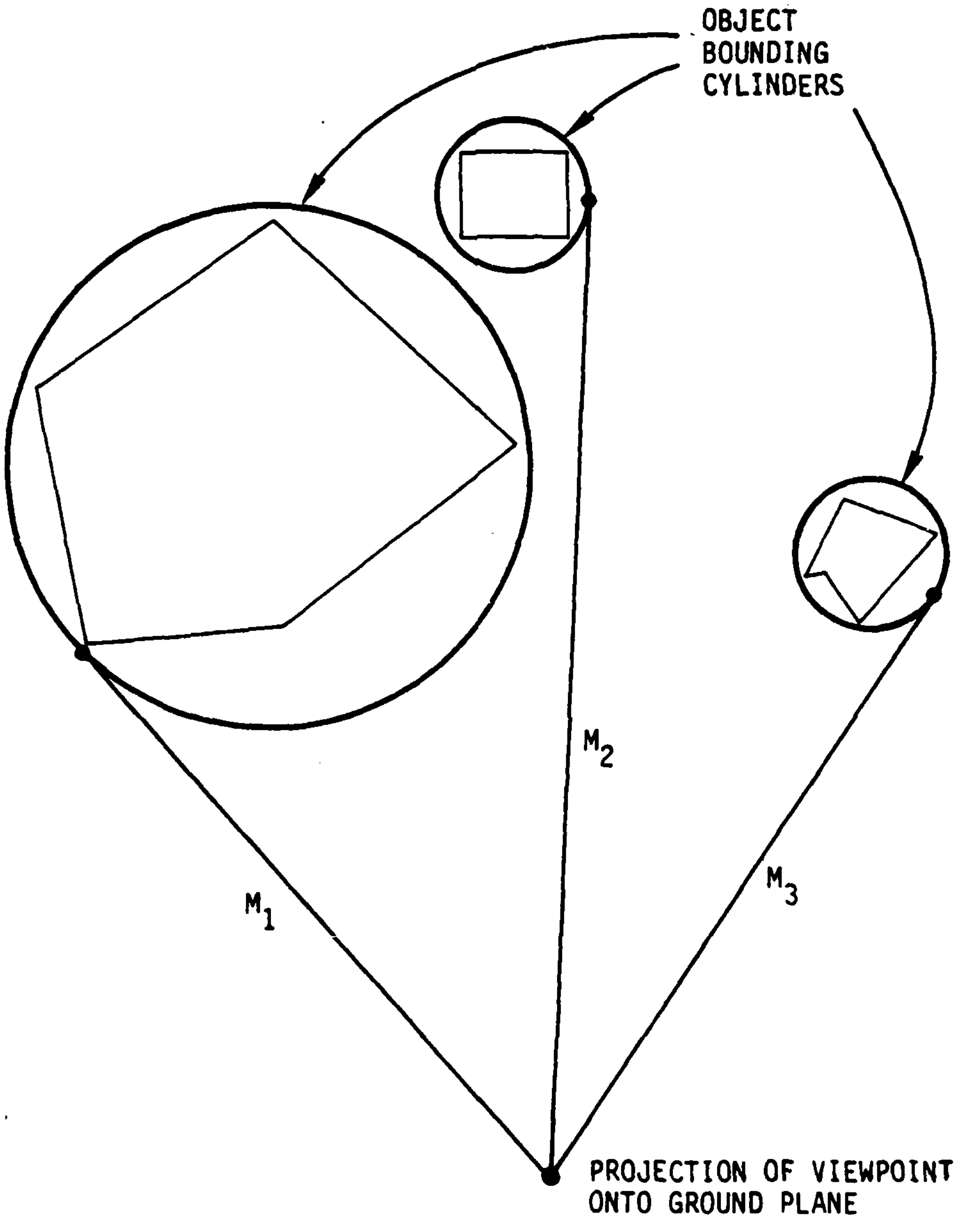


Figure 10. Ordering metric.

orderings in some situations. A slight variation of the metric, using the same object bounding cylinders used in the algorithm for listing objects in view, gave the correct ordering in all cases, if the bounding cylinders did not overlap. The new metric, illustrated in Figure 10, is the ground distance between the projection of the viewpoint onto the ground plane and the tangent point of the cylinder.

The requirement for nonoverlapping bounding cylinders is slightly more restrictive than the linear separability required by the algorithms based on separating planes. For the low-density data bases required for this project, the restrictions are not significant. Our algorithm is advantageous because it is easy to implement, fast, and requires no additional data base structure other than the bounding cylinders.

Representation

We decided to display trees as 2D rather than 3D objects to reduce the number of displayed polygons and because 2D stylized trees are more believable than attempting more realistic 3D trees using a limited number of polygons. To keep the 2D trees from looking like billboards, however, they were always rotated to "face" the viewer to give them a cone-shapped appearance. The problem was that the processing time to rotate the vertices of up to 100 trees in the FOV would have been prohibitive.

This problem was solved when we developed an object library for storing descriptions of generic objects that are replicated throughout the data base. Using the library meant that only one set of vertices had to be stored for each generic tree type; a number of orientations of each tree type could also be stored and still be well within the host computer memory limitations. Instead of computing the rotations, the real-time software had only to look up in the object library the appropriate orientation based on the aircraft heading.

As part of follow-on work to this contract, we propose to expand the concept of stored orientations to include storing multiple levels of detail for generic objects (see Recommendations section).

RECOMMENDATIONS

Short-term Enhancements

The following paragraphs list enhancements that would be straightforward to implement: modifications that could be made in a short time without purchasing additional hardware or rewriting major portions of the software.

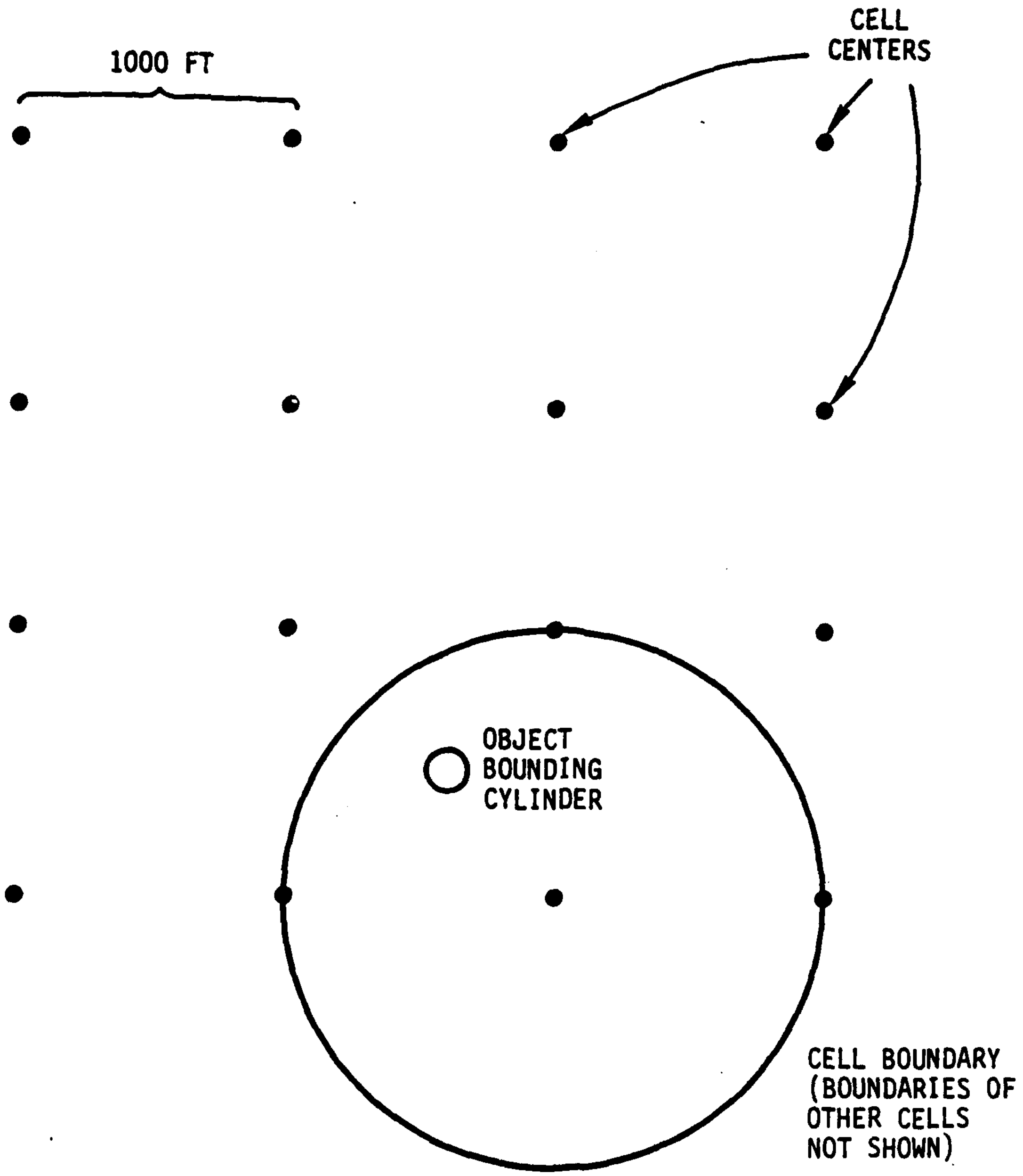


Figure 9. Data base cell structure.

Screen coordinates: A two-space system in which the upper pixel farthest left is (0,0) and the lower pixel farthest right is (511, 511).

UNIQUE ALGORITHM SOLUTIONS

System processes for which algorithm development presented problems are discussed in this section. The solutions arrived at were all unique algorithms.

Objects in View

An algorithm was needed to cull those objects within the FOV from the large number of objects in the data base to limit the processing required in subsequent stages. Early in the development phase, we arrived at the idea of sectioning the data base into square cells and doing a rough test on cell inclusion before testing individual objects. In working out the details of the algorithm, however, it turned out to be difficult to determine the set of square cells intersecting the wedge-shaped viewing area (sector).

The algorithm was simplified considerably by using circular overlapping cells rather than square ones. Using the circular cells had definite advantages: circles are simpler to deal with geometrically; the new cell structure eliminated the need to test for duplicate objects: and the test for object inclusion using the object bounding cylinders became nearly identical to the cell test, simplifying the software.

Figure 9 shows the data base cell structure.

Visibility Ordering

An algorithm was needed to determine the order in which objects were to be painted into image memory to properly remove hidden surfaces. This function was part of the "painter's algorithm" by which hidden surfaces are removed by polygon overwrite. Since the correct order of the polygons within each object is determined off line, the real-time software need only sort the objects.

Published algorithms for object ordering used the notion of separating planes and a "tree" structure that is accessed in real time. After a thorough investigation, we found that such an algorithm would be very difficult to implement, might not execute at real-time speeds, and would greatly complicate the structure of the data base.

We decided to compute an "ordering metric" by which objects could be ordered using a simple numerical sort. At first we believed we could define the metric for each object to be the distance from the viewpoint to the object's centroid. This metric, however, was shown to give incorrect

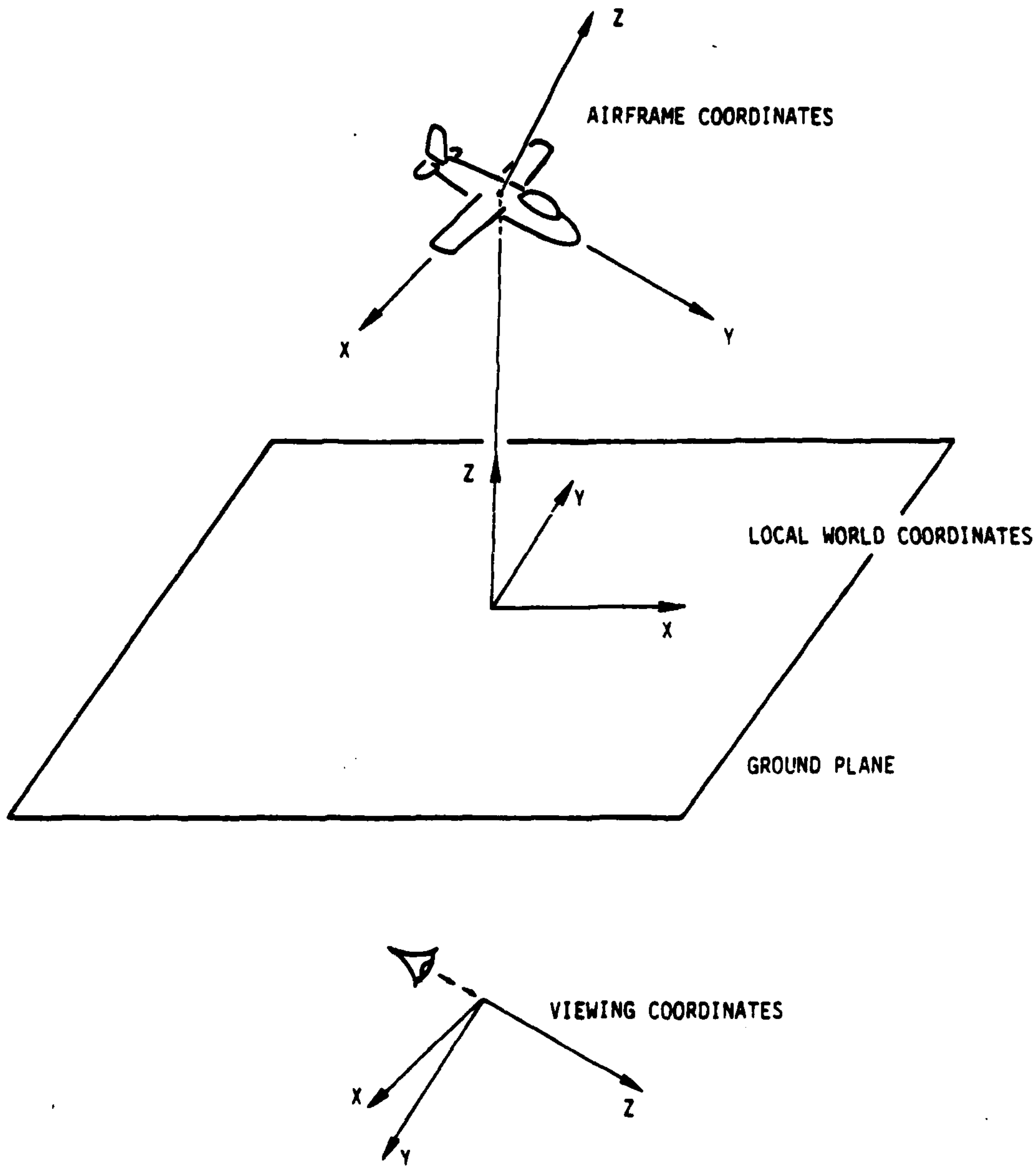


Figure 8. Local world, airframe, and viewing coordinate systems.

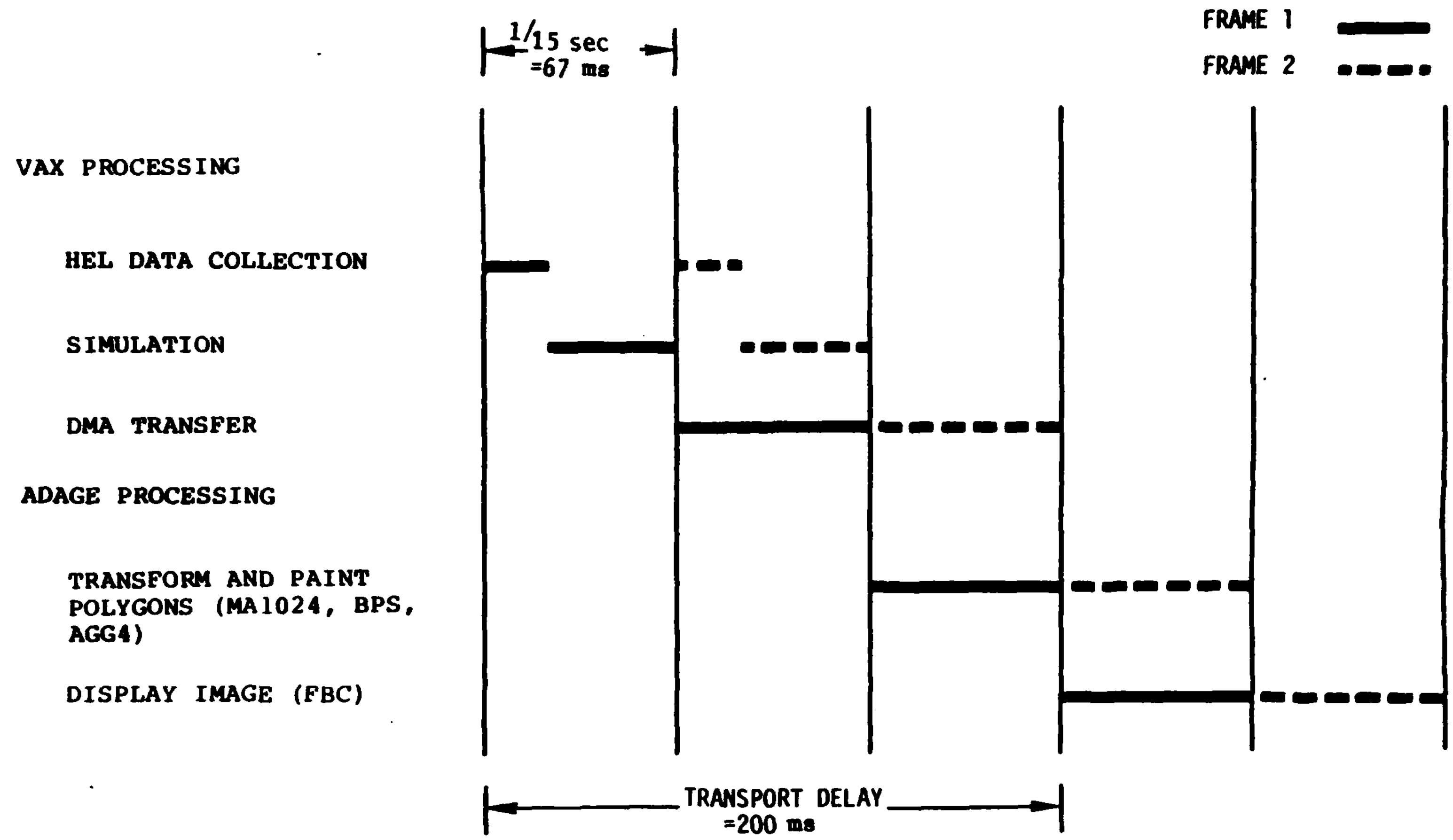


Figure 7. System timing.

VAX PROCESSING

HEL Data Collection

Simulation

- List objects
- Order objects
- Determine horizon
- Set up horizon
- Create polygon and vertex lists
- Compute coefficient matrix

DMA transfer ⇩

ADAGE PROCESSING

MA1024 and BPS

- Rotation
- Clipping
- Perspective division
- Polygon-to-trapezoid conversion

APEX

- Scan-convert trapezoid

FBC

- Display image

Figure 6. Functional breakdown of system.

Because 2.5D objects and the moving 3D object are assigned orientations in real time, the VAX must select proper orientations for these objects from the set stored in the data base. The 2.5D object orientations are chosen using the line-of-sight (LOS) vector. The moving object orientation is determined by the object's preset path.

The Adage system accepts the ordered list of polygons from the host interface, paints the polygons into image memory, and creates the output video signal.

Before being painted, the polygons are transformed to screen space through a three-step process: 1) the polygon vertices are transformed from local world coordinates to clipping coordinates, 2) the polygons are clipped to the display screen, and 3) the vertices are transformed from clipping to screen coordinates. Coordinate transformations are done by the MA1024 processor; clipping is done by the BPS processor. The BPS breaks each polygon into trapezoids that are then sent to the AGG4 processor to be painted into image memory. The frame buffer controller (FBC) creates the output video signal from the image memory.

Figure 6 is a functional breakdown of the system; Figure 7 is the system timing diagram.

Coordinate Systems

Figure 8 illustrates three of the seven coordinate systems used at various points in the processing chain.

World coordinates: Basic reference system for all simulated objects. It is a right-handed, Cartesian, three-space system with $z = 0$ as the ground plane.

Object coordinates: Same as world coordinates, except the system is defined for each object with the origin at the intersection of the bounding cylinder axis and the ground plane.

Local world coordinates: Same as world coordinates, except the origin is at the projection of the viewpoint onto the ground plane ($z = 0$).

Airframe coordinates: Same as world coordinates, except the origin is at the viewpoint and the axes are rotated so that the y-axis is along the LOS and x-axis is parallel to the bottom of the viewing window.

Viewing coordinates: Same as airframe coordinates, except the z-axis is rotated 90° such that it is along the LOS.

Clipping coordinates: Same as viewing coordinates, except that w (the "perspective coordinate") is computed; x, y, and w are scaled to allow clipping of x and y against w; and the distance from the viewpoint to the near clipping plane is subtracted from z. (See sections on the MA1024 transformation processor in the RDS 3000 User's Guide.)

DEC VAX 11/780

with floating point accelerator

Adage 3000 (IKONAS)

(AGG4) APEX	processor
BPS32	processor
(3) GM256	image memory
IK114	DMA interface
LUV024	look-up table
MA1024	processor
MCM4	memory
SR8	memory
XBS34	crossbar switch

Figure 5. System components.

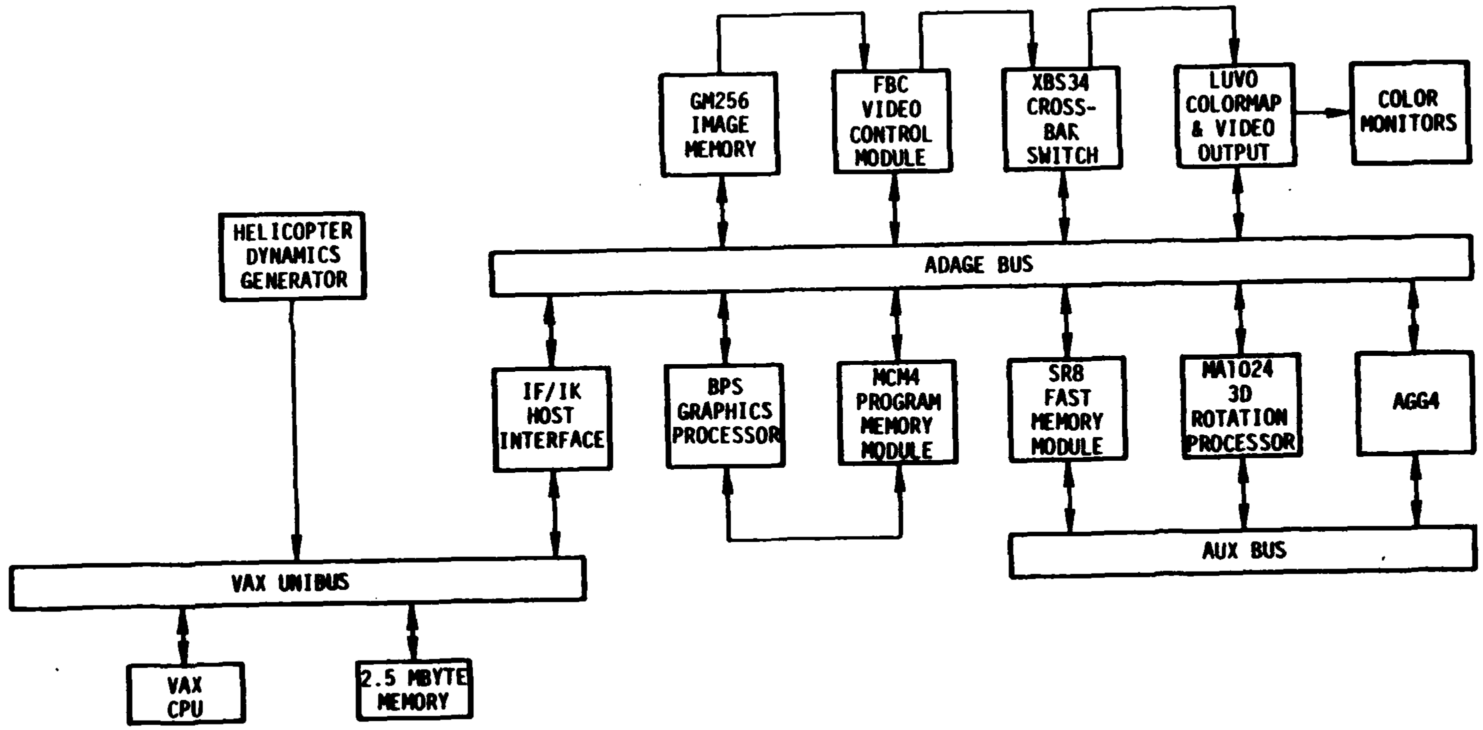


Figure 4. System hardware configuration.

SYSTEM DESCRIPTION

Off-line Processing

Off-line processing consists entirely of terrain data base creation, which involves creating and placing objects, ordering polygons within each object by visibility priority, assigning objects to cells, and enforcing data base restrictions (e.g., bounding cylinders must not overlap and maximum object density must not be exceeded).

The data base is created or modified using an editor and then converted into real-time format by a post-processor. Color data are adjusted using a palette editor; supporting data files are modified using a text editor.

Real-time Processing

The hardware for real-time processing consists of the VAX 11/780 host computer and Adage 3000 graphics system, the latter containing several processors capable of running concurrently. The Technology Service Corporation (TSC) conducted a survey of commercially available graphics systems prior to selecting the graphics hardware to be used in the simulator. Product literature from more than 20 vendors was studied, and TSC met with sales and technical representatives from many of these companies, and visited the manufacturing plants of the four top contenders. The results of the graphics system survey and selection are documented in the report, Graphics System Survey for Real-time Terrain Image Generator Project (Hardy, 1983).

Processing is pipelined through the VAX and the Adage processors. The VAX is a 32-bit processor; the Adage processors are essentially 16-bit, although some calculations can be done in 32-bit precision. Figure 4 shows the system hardware configuration, and Figure 5 lists the system components.

The VAX is the first processor in the pipeline. Its primary function is to create for each frame an ordered list of visible polygons, although it also performs some support functions such as setting up coefficient lists for coordinate transformation.

Because the visibility ordering of the polygons within each object is determined off line, much of the VAX processing may be performed at the object level. To create the polygon list, the VAX first determines the cells intersecting the viewing sector and creates a list of visible objects by testing the objects contained in these cells for intersection with the viewing sector. It then orders the objects by visibility priority, deletes back-facing polygons, and translates the polygon vertices from object coordinates to local world coordinates.

The system has a more accurate test for crash detection. For hills, the system tests for intersection of the observer with a cone contained within the hill. For smaller objects, the system tests for intersection of the observer with the bounding cylinder of the object.

Antialiasing

A method to minimize such distracting visual effects as scintillation and stairstepping had to be contained in the system. An algorithm to perform antialiasing for 2D features was developed, but time and resources did not allow for its implementation (see Recommendations section).

Haze and Fog Effects

The attenuation of light in the atmosphere caused by haze and fog was to be simulated. However, since attenuation could not be implemented in the selected graphics hardware without incurring a significant loss in performance, haze and fog effects were dropped from the project.

Software Quality

The contract required that the software for the system be documented and written in a high-level programming language where possible.

The programming language used for the VAX 11/780 was FORTRAN, and all documentation was done as the system developed. System development was based on stepwise refinement: the system design was developed first, then the algorithm design was developed from the system design, and finally the FORTRAN code was written using the system and algorithm designs as comments. By incorporating the system and algorithm designs into the FORTRAN code, the documentation remained current and easy to generate.

A program was written to identify the system design comments in the code and to separate them from it.

The FORTRAN code is indented and made more readable by another program. Coding conventions used in the project are in the appendix.

The programming language used for the Adage BPS is a C-based compiler language developed at the University of South Carolina. The compiler, GIA, creates microcode that is assembled and loaded into the BPS.

Microcode was written for the Adage AGG4 because it is the only language available for that processor.

Update Rate

The update rate is the number of frames displayed per second (fps). Each frame is a complete scene that includes all objects within the FOV.

The update rate required was originally 16 fps which was later changed to 15 fps to conform to the video monitor's field rate. The system was delivered with a field rate of 15 fps.

If the scene contains a higher object density than allowed, the system reduces the update rate and displays a message to the control monitor.

Image Resolution

Image resolution is the number of pixels (picture elements) displayed on the screen. We delivered the 512 x 512 image resolution required by the contract.

Occultation

Occultation is the process of not displaying objects hidden partly or entirely from the viewer. This process was required to be a dynamic process achieved in real time. The system delivered performs occultation without error.

Hidden Surface Removal

Hidden surfaces are those facing away from the viewer (not to be confused with occulted surfaces, which face the viewer but are hidden). Since the viewer cannot see these surfaces, they should not be displayed. The system removes all hidden surfaces in real time, as required.

Number of Moving Objects

One independently moving vehicle was required within the gaming area. The system delivered has 20 independently moving vehicles

Special Features

The contract required the system to have three special features: crash detection, antialiasing, and haze and fog effects.

Crash Detection

The system was required to compute the intersection of the observer with the center of gravity of each object to determine whether the observer was crashing into that object.

Long-term Enhancements

The following items, recommended as long-term enhancements, would require major changes to the simulator.

More displayed polygons/faster update rate: Because we are near the maximum polygon capacity of the Adage 3000 with the current simulator, any substantial increase in performance would hinge upon Adage improving the 3000 system or the use of another graphics system. It is already apparent that much higher performance special-purpose systems will be available within the next several years. We are now investigating one recently announced system, the GTI Poly 2000, that shows considerable promise. Increasing the polygon capacity would probably also entail restructuring the VAX real-time software or off-loading the VAX processing to another processor.

Off-loading VAX processing: With the rapid advances being made in minicomputer technology, it may be desirable to move some or all of the VAX processing onto a less expensive and/or faster minicomputer, to free up VAX resources or increase simulator performance. Off-loading would require major changes to the software.

Increased display resolution: The display resolution of the system could be increased from 512 x 512 to 1024 x 1024 by purchasing several additional memory boards for the Adage 3000 and making a moderate-sized change to the software. We estimate that because the graphics processor must write additional pixels, increased resolution would entail a decrease in polygon capacity or, equivalently, a slowdown in update rate by a factor of two to four.

Rolling terrain: This capability is the display of facets fitted to a varying-height terrain model and is currently available only in very high-fidelity simulators such as the Evans and Sutherland CTS. To add rolling terrain to the HELICON system would require substantial algorithm development at least, and probably the incorporation of more powerful graphics hardware.

Atmospheric attenuation: To portray attenuation accurately, the simulator would have to calculate pixel colors pixel by pixel which is far too much processing to be handled by the hardware currently used. To add attenuation to the system, we would need to either develop special-purpose hardware or identify a low-cost graphics system with this capability. We know of no such system now available, although GTI has indicated plans to incorporate attenuation into their Poly 2000 system.

REFERENCES

1. Gupta, S., & Sproull, R. F. (1981). Filtering edges for gray-scale displays. SIGGRAPH Conference Proceedings, 15 (3), 1-5.

2. Hardy, J. G. (1983). Graphics system survey for real-time terrain image generator project. Santa Monica, CA: Technology Service Corporation.

3. Ikonas Graphics System, Inc. (1982). RDS 3000 user's guide. Billerica, MA: Author.

APPENDIX

FORTRAN CODING CONVENTIONS

FORTRAN CODING CONVENTIONS

INTRODUCTION

The purpose of the coding conventions is to make the software more readable and uniform, which should in turn make it easier to verify and maintain. The intent of the information here is not to promote a set of fixed rules, but to establish guidelines to serve as a general template.

PROGRAM STRUCTURE

Each process is generally coded as a subroutine that makes up one source file. The following guidelines, for the most part, apply to process subroutines; utility subroutines may be free form.

Subroutine Blocks

Each source file (subroutine) consists of the elements listed below. These occur in the source file in the order given, except that PDL (Program Design Language) and FORTRAN lines are intermixed.

- Subroutine statement: The parameter list contains a flag, ERROR, that is returned .TRUE. if an error occurs during execution of the subroutine. All other subroutine input/output (I/O) is via common blocks.

- Process descriptions: These are sets of comment lines, marked by "cs" in columns 1 and 2, that briefly describe the process and its I/O. Any subprocesses are also described here.

- Common blocks: The source code for each common block is contained in its own file, which is brought into the subroutine source file using the INCLUDE statement. A common block file is made up of a description of the common block, variable declarations, and the COMMON statement.

- Local variable declarations: All variables are declared if the variable type lists the number of bytes explicitly (e.g. INTEGER*4 instead of INTEGER). Each declaration is followed by a comment (on the same line) that describes the variable.

- PDL lines: These define the process algorithm and appear in the FORTRAN source file as all-lowercase comment lines marked by "cp" in columns 1 and 2. PDL lines are intermixed with FORTRAN executable statements.

- **FORTRAN executable statements**: Standard FORTRAN 77 is used.

- **Footnotes**: These are special PDL lines placed at the end of the source file. Numbered footnotes give additonal information about the algorithm (e.g., equations). Footnotes referred to by letters contain software developement notes.

The FORTRAN source files are the ultimate source for all VAX process descriptions and PDL. All changes to the FORTRAN files thus "bubble up" to the system.

General Comment Lines

Comment lines that are not PDL or part of a process description are marked by "c" in column 1 and a blank in column 2.

Blank Lines

Spaces are used to bracket binary operators, the assignment operator ("="), and most nonsubscript parentheses. A space is usually inserted after a comma. The following examples show the conventions for using spaces in various types of statements:

```
A = ( B + C ) * D(I)
CALL SUB ( A, B, C )
OPEN ( FILE = FILE, UNIT = UNIT )
IF ( A .EQ.  B ) THEN
READ (1,*, ERR=8000) X
```

Indenting

PDL lines and FORTRAN statements that occur within control structures (IF THEN ... ELSE ... END IF; DO and DO WHILE loops) are indented four spaces. Indenting is cumulative for nested control structures.

Form Feeds

Form feeds are inserted in the source file to cause page ejections after the process description, the common blocks, and the local variable declarations.

Continuation Character

The continuation character is "_".

Variable Names

The general variable names are as descriptive as possible without being too long (maximum is 15 characters). Other guidelines are:

- Since all variable types are declared, a variable name may start with any letter.
- For cases in which a common block variable may be confused with similar variables in other common blocks, the variable name starts with the common block name or an abbreviation of the common block name.
- Location variables should contain the name of the coordinate system assumed for that variable followed by the coordinate (X, Y, or Z), as in HELWORLDX and OBJLOCALZ. The possible coordinate systems used in VA processes are world, object, and local.
- Variable names contain no more than two underscores.

Statement Numbers

Because of the DO ... END DO and IF ... END IF constructs available in FORTRAN 77, it is expected that statement numbers will rarely be used except as error exits out of read/write statements and the like. Executable statements have four-digit statement numbers starting with 1000. Error exits have numbers starting with 8000. If the RETURN statement is numbered, it is given 9000. FORMAT statements have two-digit numbers starting with 10.

REPRODUCED AT GOVERNMENT EXPENSE

END

FILMED

10-85

DTIC